Small sample solutions for SEM

Yves Rosseel
Department of Data Analysis
Ghent University – Belgium

Meeting of the SEM Working Group
Tilburg, 10 March 2022

structure of this talk

- small sample problems for structural equation modeling (SEM)
- small sample solution 1: bounded estimation (avoid convergence)
  - joint work with my PhD student Julie De Jonckere
- small sample solution 2: bias-correction
  - joint work with my PhD student Sara Dhaene
- small sample solution 3: structural after measurement (SAM)
  - joint work with Wen Wei Loh (postdoc UGent)
- last slide

slides available at `http://lavaan.org`

small sample problems in SEM (in the frequentist framework)

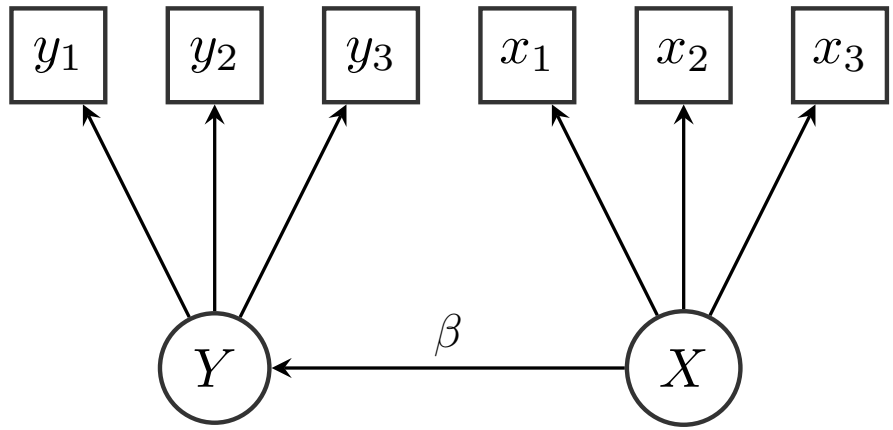
- the statistical machinery behind SEM is based on large sample theory
- in practice, (very) small sample sizes ($N < 100$) are simply a reality
  - population is (very) small (i.e., children with severe facial burns)
  - high cost/effort per observation (i.e., fMRI data)
- this leads to many issues:
  - nonconvergence (the optimizer cannot find a solution)
  - extreme/nonsensical parameter values
  - parameter bias
  - standard errors, confidence intervals, test statistics cannot be trusted
- two problematic settings:
  - small models, (very) small sample sizes (e.g., $N = 20$)
  - large models, (relatively) small sample sizes (e.g., $N = 150$)

small sample solutions for SEM

- the most popular solution is to switch to a Bayesian approach
  - major advantage: no need for large sample asymptotics
  - correct standard errors and credible intervals even in small samples
  - but model evaluation requires a new set of skills
  - you need a prior distribution for each (free) parameter
  - when the sample size is (very) small (say, $N < 50$), Bayesian estimation only works with (highly) informative priors (otherwise, mode-switching issues may occur)
- in our SEM lab in Ghent, we try to find solutions for the frequentist framework
  - understanding and avoiding convergence issues (very small samples)
  - two-step approaches (factor score regression, SAM)
  - improving the quality of estimators (in small samples)
  - noniterative estimators

problematic setting 1: very small samples

- consider the following SEM:



- this is a small model, with only 13 free parameters:
  - the factor loadings are set to 1, 0.8 and 0.6; all variances are set to 1.0
  - the regression coefficient is set to $\beta = 0.25$
  - (very) low reliabilities for the indicators: 0.265–0.515
- from this population model, we will generate a small sample ($N = 20$)

data generation ($N = 20$)

```
> library(lavaan)
> pop.model <- '
+     # factor loadings
+     Y =~ 1*y1 + 0.8*y2 + 0.6*y3
+     X =~ 1*x1 + 0.8*x2 + 0.6*x3
+
+     # regression part
+     Y ~ 0.25*X
+ '
> set.seed(8)
> Data <- simulateData(pop.model, sample.nobs = 20L)
```

fitting the model using ML

```
> model <- '
+     # factor loadings
+     Y =~ y1 + y2 + y3
+     X =~ x1 + x2 + x3
+
+     # regression part
+     Y ~ X
+ '
> fit <- sem(model, data = Data, estimator = "ML")

lavaan WARNING: the optimizer warns that a solution has NOT been found!
```

lavaan output

```
> parameterEstimates(fit, header = FALSE, ci = FALSE, output = "text")[1:13,]

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  Y =~
    y1                1.000
    y2                1.683       NA
    y3                1.051       NA
  X =~
    x1                1.000
    x2              268.924       NA
    x3                0.428       NA

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  Y ~
    X                -0.159       NA

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                1.706       NA
   .y2                0.763       NA
   .y3                1.066       NA
   .x1                1.408       NA
   .x2             -368.917       NA
   .x3                1.551       NA
```

R = 10000 replications: number of nonconverged solutions

sample size	no-bounds
10	5113
15	3919
20	2971
25	2237
30	1668
40	929
50	521
60	264
70	166
80	73
90	37
100	21

solution: 'bounded estimation'

- by default, lavaan (and most SEM software) uses unconstrained estimation (using quasi-Newton methods)
- given the data, we can determine data–driven lower and upper bounds for a subset of the model parameters (variances, covariances, and factor loadings)
- for more details, see

  > De Jonckere & Rosseel (2022). Using bounded estimation to avoid nonconvergence in small sample structural equation modeling. DOI: `https://doi.org/10.1080/10705511.2021.1982716`

- works for all estimators, missing values, categorical, multigroup, ...
- we assume $N > P$ (P is the number of observed variables) and the model is identified
- lavaan limitation: does not work (out of the box) in the presence of linear equality constraints

lavaan: adding lower and upper bounds manually

```
> model.bounds <- '
+   # factor loadings
+     Y =~ y1 + y2 + y3
+     X =~ x1 + upper(5)*x2 + x3
+
+   # residual variances
+     x2 ~~ lower(0)*x2
+
+   # regression part
+     Y ~ X
+ '
> fit.bounds <- sem(model.bounds, data = Data, estimator = "ML")
```

lavaan output with manual bounds

```
> parameterEstimates(fit.bounds, header = FALSE, ci = FALSE, output = "text")[1:1

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  Y =~
    y1                1.000
    y2                1.410    1.056    1.335    0.182
    y3                0.927    0.615    1.507    0.132
  X =~
    x1                1.000
    x2                1.711    0.897    1.907    0.056
    x3                0.511    0.305    1.675    0.094

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  Y ~
    X                -0.035    0.230   -0.153    0.878

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .x2        (lb)    0.000    1.151    0.000
   .y1                1.583    0.642    2.464    0.014
   .y2                0.897    0.848    1.058    0.290
   .y3                1.070    0.484    2.213    0.027
   .x1                0.606    0.437    1.385    0.166
   .x3                1.342    0.437    3.074    0.002
```

finding (data–driven) lower and upper bounds automatically

- in scalar notation, we can write the one-factor model as

$$y_p = \lambda_p \, f + \epsilon_p$$

- we assume $\text{Cov}(f, \epsilon_p) = 0$ and write $\text{Var}(\epsilon_p) = \theta_p$ and $\text{Var}(f) = \psi$, and therefore

$$\text{Var}(y_p) = \lambda_p^2 \, \psi + \theta_p$$

- the idea is simple: we should put lower/upper bounds on the parameters λ_p, ψ and θ_p so that the right hand side stays nonnegative and never exceeds $\text{Var}(y_p)$

- in practice, $\text{Var}(y_p)$ is replaced by the observed sample variance of y_p (that is why the bounds are data–driven)

- for θ_p, it is clear that the 'standard' lower bound is zero, and the 'standard' upper bound is $\text{Var}(y_p)$

finding (data–driven) lower and upper bounds automatically (2)

- we fix the metric of the factor f by fixing the first factor loading to 1
- the upper positive bound for λ_p is given by

$$\lambda_p^{(u)} = \sqrt{\frac{\text{Var}(y_p)}{\psi^{(l)}}}$$

  where $\psi^{(l)}$ is the lower bound for the variance of the factor
- the lower bound for the factor variance can be expressed as:

$$\psi^{(l)} = \text{Var}(y_1) - [1 - \text{REL}(y_1)]\text{Var}(y_1)$$

  where $\text{REL}(y_1)$ is the (unknown) minimum reliability of the first (marker) indicator y_1
- we will often assume that $\text{REL}(y_1) \geq 0.1$

lavaan: using 'standard' bounds

```
> fit.stan <- sem(model, data = Data, estimator = "ML", bounds = "standard")
> parTable(fit.stan)[,c("lhs", "op", "rhs", "free", "lower", "upper", "est")]
```

	lhs	op	rhs	free	lower	upper	est
1	Y	=~	y1	0	1.000	1.000	1.000
2	Y	=~	y2	1	-3.074	3.074	1.410
3	Y	=~	y3	2	-2.692	2.692	0.927
4	X	=~	x1	0	1.000	1.000	1.000
5	X	=~	x2	3	-4.089	4.089	1.711
6	X	=~	x3	4	-3.315	3.315	0.511
7	Y	~	X	5	-Inf	Inf	-0.035
8	y1	~~	y1	6	0.000	1.941	1.583
9	y2	~~	y2	7	0.000	2.037	0.897
10	y3	~~	y3	8	0.000	1.563	1.070
11	x1	~~	x1	9	0.000	1.271	0.606
12	x2	~~	x2	10	0.000	2.362	0.000
13	x3	~~	x3	11	0.000	1.552	1.342
14	Y	~~	Y	12	0.005	2.156	0.572
15	X	~~	X	13	0.141	1.413	0.807

'wide' bounds

- suppose the lower/upper bounds for a parameter θ are (0, 10)
- we can increase the upper bound with, say, 10%: (0, 11)
- similarly, we can decrease the lower bound with 10%: (-1,11)
- why should we use 'wide' bounds:
  - more 'wiggle' room for the optimizer
  - somewhat better convergence rates
  - we allow for (small) negative variances (alerting the user)
- currently, the 'best' (across many models/datasets) choice seems to be:
  - minimum reliability first indicator: 0.1 (or higher)
  - minimum residual variance latent variable: 0.005
  - increase/decrease bounds of observed variances with a factor 1.20/1.05
  - increase/decrease bounds of factor loadings with a factor 1.10/1.10
  - increase upper bounds of latent variances with a factor 1.30

lavaan: using the 'optim.bounds' argument

```
> fit.wide <- sem(model, data = Data, estimator = "ML",
+                 optim.bounds = list(upper = c("ov.var", "lv.var", "loadings"),
+                                     lower = c("ov.var", "lv.var", "loadings"),
+                                     upper.factor = c(1.20, 1.30, 1.10),
+                                     lower.factor = c(1.05, 1.00, 1.10),
+                                     min.var.lv.endo = 0.005,
+                                     min.reliability.marker = 0.1))
```

lavaan: using 'wide' bounds

```
> fit.wide <- sem(model, data = Data, estimator = "ML", bounds = "wide")
> parTable(fit.wide)[,c("lhs", "op", "rhs", "free", "lower", "upper", "est")]
```

	lhs	op	rhs	free	lower	upper	est
1	Y	=~	y1	0	1.000	1.000	1.000
2	Y	=~	y2	1	-3.689	3.689	1.400
3	Y	=~	y3	2	-3.231	3.231	0.938
4	X	=~	x1	0	1.000	1.000	1.000
5	X	=~	x2	3	-4.907	4.907	1.787
6	X	=~	x3	4	-3.978	3.978	0.528
7	Y	~	X	5	-Inf	Inf	-0.055
8	y1	~~	y1	6	-0.097	2.329	1.584
9	y2	~~	y2	7	-0.102	2.445	0.915
10	y3	~~	y3	8	-0.078	1.875	1.059
11	x1	~~	x1	9	-0.064	1.526	0.636
12	x2	~~	x2	10	-0.118	2.834	-0.118
13	x3	~~	x3	11	-0.078	1.863	1.336
14	Y	~~	Y	12	0.005	2.802	0.570
15	X	~~	X	13	0.141	1.794	0.777

lavaan output with wide bounds

```
> parameterEstimates(fit.wide, header = FALSE, ci = FALSE, output = "text")[1:13,

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  Y =~
    y1                1.000
    y2                1.400    1.039    1.347    0.178
    y3                0.938    0.623    1.505    0.132
  X =~
    x1                1.000
    x2                1.787    0.932    1.918    0.055
    x3                0.528    0.305    1.733    0.083

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  Y ~
    X                -0.055    0.230   -0.239    0.811

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                1.584    0.640    2.474    0.013
   .y2                0.915    0.832    1.100    0.271
   .y3                1.059    0.485    2.184    0.029
   .x1                0.636    0.424    1.498    0.134
   .x2        (lb)   -0.118    1.194   -0.099
   .x3                1.336    0.435    3.071    0.002
```

how well does it work?

- simulation 1 (from the paper): using the simple model with 13 parameters:
- four settings:
  - no bounds
  - ov.var: lower zero bounds for (residual) variances of observed indicators only
  - standard bounds: lower and upper bounds for variances and factor loadings
  - wide bounds: somewhat wider bounds (allowing for negative variances, more 'wiggle room' for the optimizer)

R = 10000 replications: number of nonconverged solutions

sample size	no-bounds	ov.var	standard	wide
10	5113	921	13	0
15	3919	414	13	1
20	2971	233	14	0
25	2237	135	11	0
30	1668	66	5	0
40	929	27	4	0
50	521	5	4	0
60	264	1	8	0
70	166	0	5	0
80	73	0	2	0
90	37	0	2	0
100	21	0	1	0

summary 'bounded estimation'

- 'bounded estimation' seems to be very effective in avoiding nonconvergence (and inadmissible solutions) when the sample size is (very) small
- no (adverse) impact on unbounded parameters
- if you have a model/data combination that still doesn't converge (despite using bounds), please let us know
- future research:
  - 'fix' the noisy sample (co)variance matrix by shrinking towards a model-based 'ideal' (co)variance matrix
  - predict apriori when a model/data combination will lead to nonconvergence (and understand why)
- 'bounded estimation' allows for the use of resampling techniques even when the sample size is very small

problem 2: finite sample bias

- ML estimates are consistent, but in (small) samples, there may be some bias
- how large is this bias?
  - not much literature
  - both upwards and downwards bias
  - means/intercepts, factor loadings and regression coefficients show almost no bias, if the reliability of the indicators is high
  - recall: in the linear mixed model, the 'fixed effects' (the regression coefficients) are unbiased (Demidenko, 2013, p. 137)
- should we attempt to correct for this bias? (bias-variance trade-off)
- in the world of linear mixed models, restricted maximum likelihood (REML) estimation is widely used
- but we need something that works for all SEMs

two approaches

- recent paper:

  Ozenne, B., Fisher, P.M., & Budtz-Jørgensen, E. (2020). Small sample corrections for Wald tests in latent variable models. *Journal of the Royal Statistical Society: Series C (Applied Statistics), 69 (4)*, 841–861.

  - analytic solution
  - bias-correction, but also a small-sample correction for the Wald test

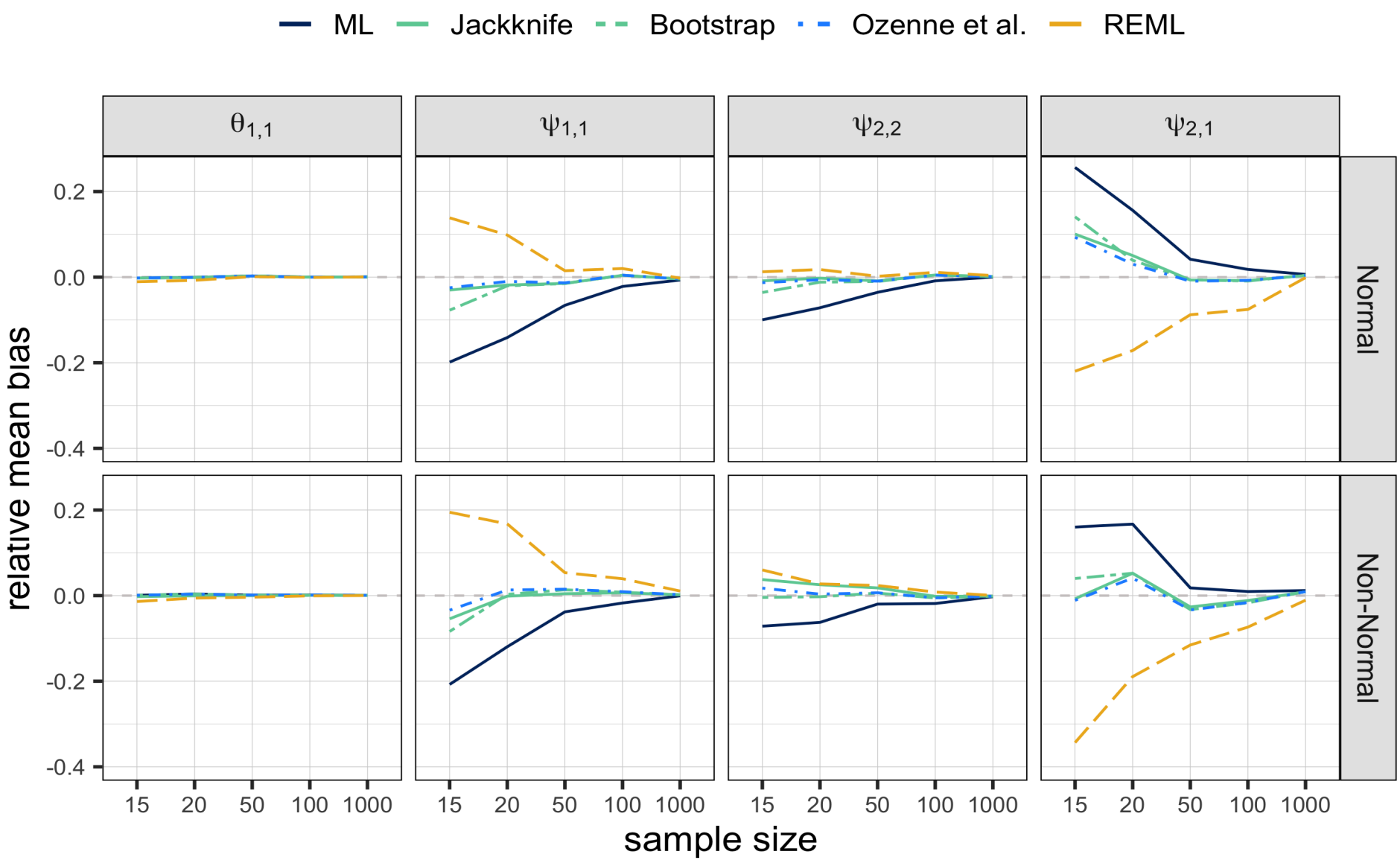
- using a resampling approach:

  Dhaene, S., & Rosseel, Y. (revision). Resampling based bias correction for small sample SEM.

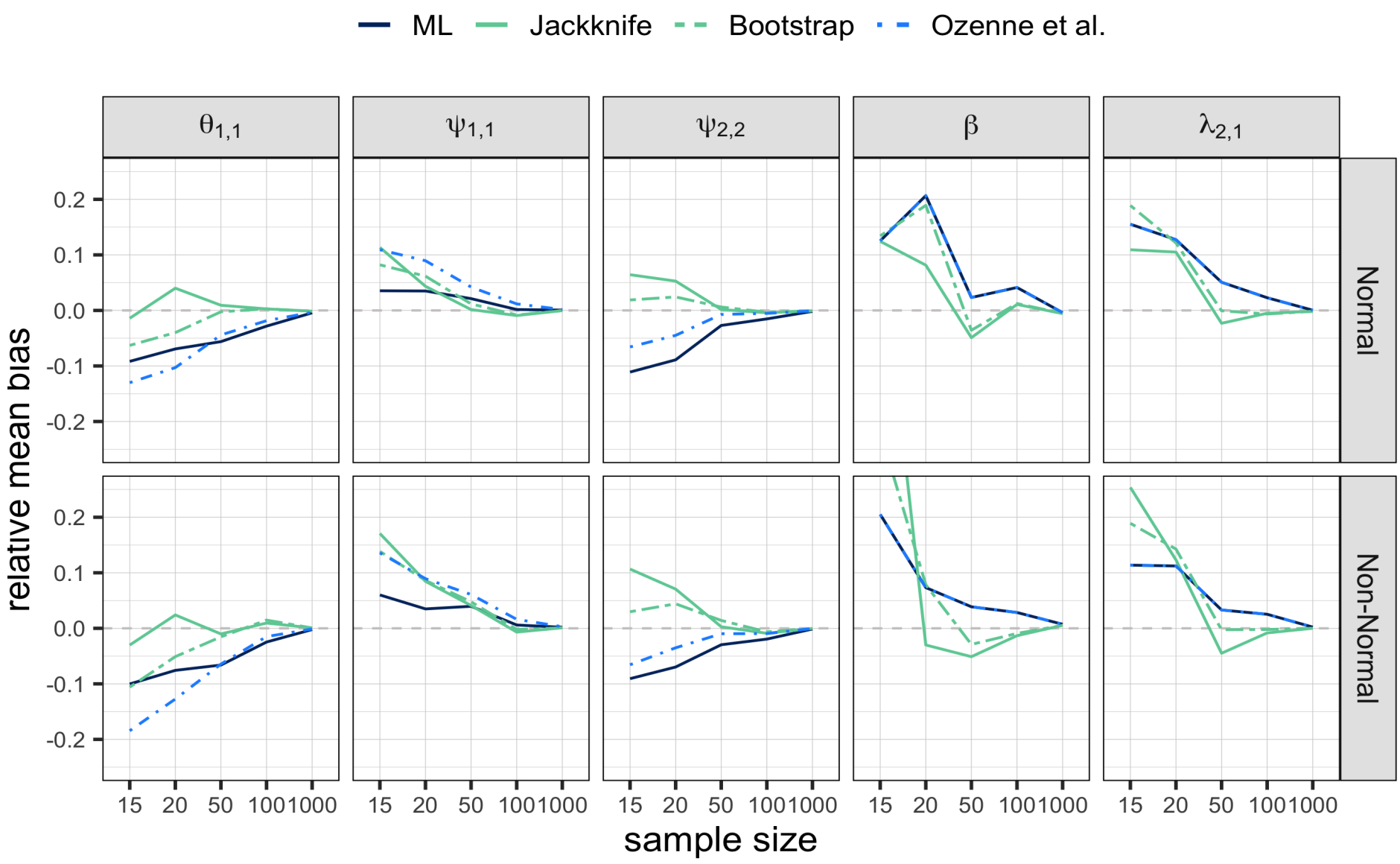
  - using the jacknife and the bootstrap to correct for small sample bias
  - simulation study comparing to Ozenne et.al. and REML (when appropriate)

- next two plots: relative mean bias when the 'indicator' reliability is (only) about 0.50

small sample bias: linear growth curve model



small sample bias: 2-factor SEM model (500 bootstrap samples)

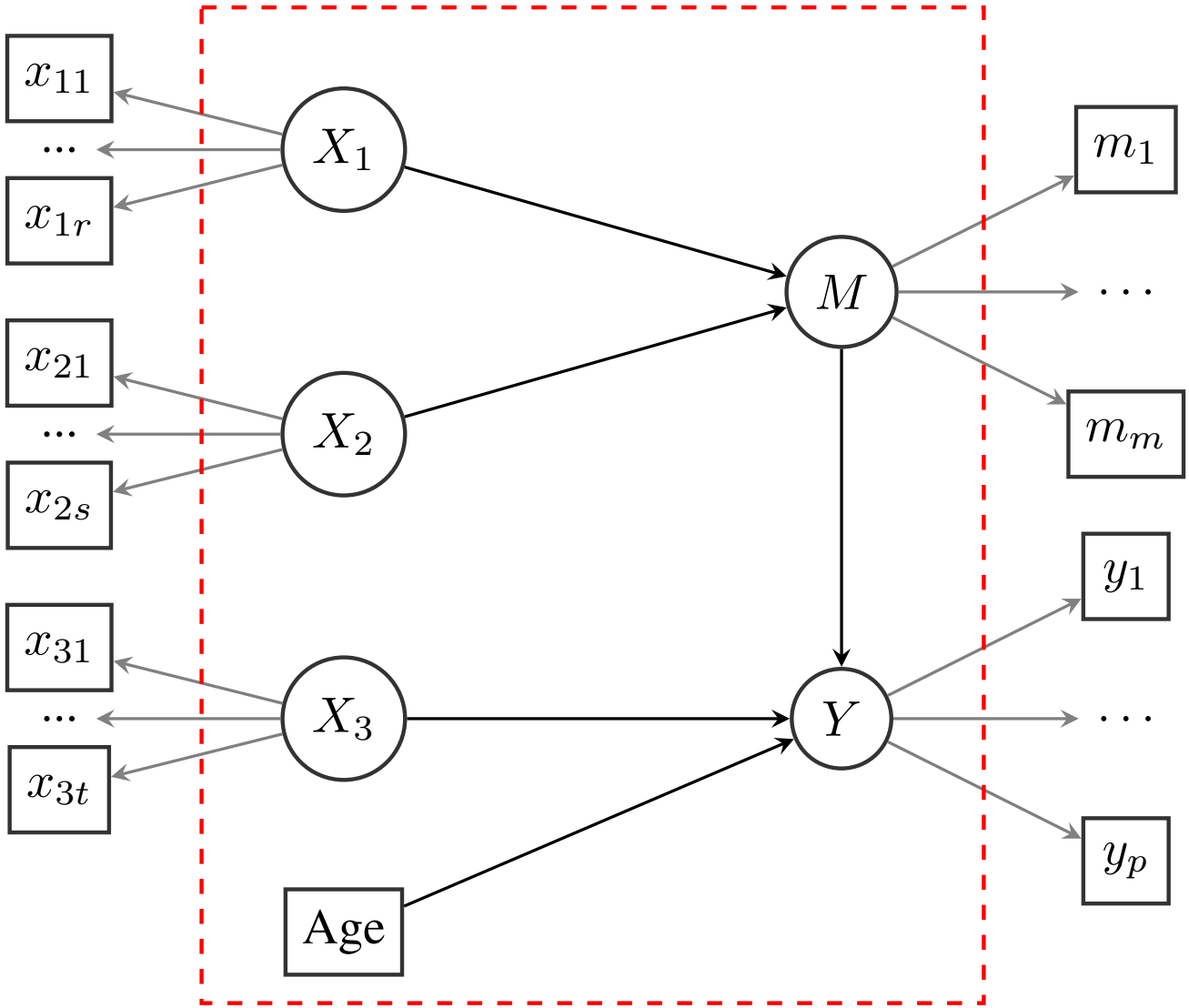


problematic setting 2: large models, small/medium samples

- 'large' models with many (say > 100) parameters, but the sample size is relatively small (say $N = 150$)
- using (traditional) SEM using ML (system-wide estimation)
  - convergence issues, unstable estimation, extreme parameter values
  - very sensitive to (local) model misspecification
- proposed solution:

  Rosseel & Loh (accepted). Structural After Measurement (SAM) approach to SEM. Available from `https://osf.io/pekbm/`
- long–standing idea: decouple estimation of the measurement part and the structural part
- local SAM is a generalization of bias-corrected factor score regression
- global SAM (not discussed in this talk) can be used in settings where local SAM cannot be used

structural versus measurement



local SAM: rationale

- the measurement model:

$$\boldsymbol{y} = \boldsymbol{\nu} + \boldsymbol{\Lambda}\boldsymbol{\eta} + \boldsymbol{\epsilon}$$

- to solve this for $\boldsymbol{\eta}$, we proceed as follows:

$$\begin{aligned} \boldsymbol{\nu} + \boldsymbol{\Lambda}\,\boldsymbol{\eta} + \boldsymbol{\epsilon} &= \boldsymbol{y} \\ \boldsymbol{\Lambda}\,\boldsymbol{\eta} &= \boldsymbol{y} - \boldsymbol{\nu} - \boldsymbol{\epsilon} \\ \mathbf{M}\boldsymbol{\Lambda}\,\boldsymbol{\eta} &= \mathbf{M}\left[\boldsymbol{y} - \boldsymbol{\nu} - \boldsymbol{\epsilon}\right] \\ \boldsymbol{\eta} &= \mathbf{M}\left[\boldsymbol{y} - \boldsymbol{\nu} - \boldsymbol{\epsilon}\right] \end{aligned}$$

  where $\mathbf{M}$ is $M \times P$ mapping matrix such that $\mathbf{M}\boldsymbol{\Lambda} = \mathbf{I}_M$

- we assume $\mathrm{E}(\boldsymbol{\epsilon}) = \mathbf{0}$ and write $\mathrm{Var}(\boldsymbol{\epsilon}) = \boldsymbol{\Theta}$; it follows that

$$\begin{aligned} \mathrm{E}(\boldsymbol{\eta}) &= \mathbf{M}\left[\mathrm{E}(\boldsymbol{y}) - \boldsymbol{\nu}\right] \\ \mathrm{Var}(\boldsymbol{\eta}) &= \mathbf{M}\left[\mathrm{Var}(\boldsymbol{y}) - \boldsymbol{\Theta}\right]\mathbf{M}^T \end{aligned}$$

local SAM: two–stage estimation

- first stage: estimation of the measurement part of the model (only); this results in estimates for $\boldsymbol{\nu}$, $\boldsymbol{\Lambda}$ and $\boldsymbol{\Theta}$, collected in $\boldsymbol{\theta}_1$; three options:
  - single CFA, multiple CFAs, as many CFAs as latent variables
- three possible solutions for the mapping matrix $\mathbf{M}$:

$$\mathbf{M} = (\boldsymbol{\Lambda}^T \boldsymbol{\Theta}^{-1} \boldsymbol{\Lambda})^{-1} \boldsymbol{\Lambda}^T \boldsymbol{\Theta}^{-1} \qquad (ML)$$

$$\mathbf{M} = (\boldsymbol{\Lambda}^T \mathbf{S}^{-1} \boldsymbol{\Lambda})^{-1} \boldsymbol{\Lambda}^T \mathbf{S}^{-1} \qquad (GLS)$$

$$\mathbf{M} = (\boldsymbol{\Lambda}^T \boldsymbol{\Lambda})^{-1} \boldsymbol{\Lambda}^T = \boldsymbol{\Lambda}^+ \qquad (ULS)$$

- we estimate $\mathrm{E}(\boldsymbol{\eta})$ and $\mathrm{Var}(\boldsymbol{\eta})$ as follows:

$$\widehat{\mathrm{E}(\boldsymbol{\eta})} = \hat{\mathbf{M}} \left[\bar{\boldsymbol{y}} - \hat{\boldsymbol{\nu}}\right]$$

$$\widehat{\mathrm{Var}(\boldsymbol{\eta})} = \hat{\mathbf{M}} \left[\mathbf{S} - \hat{\boldsymbol{\Theta}}\right] \hat{\mathbf{M}}^T$$

- second stage: $\widehat{\mathrm{E}(\boldsymbol{\eta})}$ and $\widehat{\mathrm{Var}(\boldsymbol{\eta})}$ are used to estimate $\boldsymbol{\theta}_2$, the parameters related to the structural part of the model

advantages of (local) SAM

- works equally well as "system-wide" ML when the model is correct
- more robust against (local) misspecifications
- decoupling measurement and structural part: different (noniterative) estimators can be used for each (sub)model
- extends to the multigroup and multilevel setting
- twostep-corrected standard errors, local and global fit measures are available

(current) limitations

- no support (yet) for higher-order factor models
- the 'Lambda' matrix of factor loadings cannot be rank deficient
- Var($\boldsymbol{\eta}$) must be unrestricted (no apriori zeroes/constants)

special cases of local SAM

- sum scores (biased)
- sum scores + reliability (= single-indicator models)
- factor scores (biased)
- factor scores + Croon's correction
- (linear) measurement error models (Fuller, 1987)
- two-stage method-of-moments estimator (Wall & Amemiya, 2000)
- ...

example: generate population data mediation model

```
> pop.model <- '
+     # factor loadings
+     Y =~ 1*y1 + 1.2*y2 + 0.8*y3 + 0.7*y4
+     M =~ 1*m1 + 0.5*m2 + 0.5*m3 + 0.9*m4
+     X =~ 1*x1 + 0.7*x2 + 0.6*x3 + 1.2*x4
+
+     # (co)variances
+     Y ~~ 0.5*Y; M ~~ 1.2*M; X ~~ 0.8*X
+     M ~~ 0.2*X
+
+     # regression part
+     Y ~  0.25*X + 0.40*M
+     M ~ -0.30*X
+ '
> set.seed(1234)
> Data <- simulateData(pop.model, sample.nobs = 150L, empirical = FALSE)
```

fit model using local sam

```
> model <- '
+     # factor loadings
+     Y =~ y1 + y2 + y3 + y4
+     M =~ m1 + m2 + m3 + m4
+     X =~ x1 + x2 + x3 + x4
+
+     # regression part
+     Y ~ M # direct effect of X is missing
+     M ~ X
+ '
> fit.lsam <- sam(model      = model,
+                 data       = Data,
+                 sam.method = "local",
+                 mm.args    = list(estimator = "ML"),
+                 struc.args = list(fixed.x = FALSE))
```

lavaan output

```
> summary(fit.lsam)

This is lavaan 0.6-10 -- using the SAM approach to SEM

  SAM method                                      LOCAL
  Mapping matrix M method                            ML
  Number of measurement blocks                        3
  Estimator measurement part                         ML
  Estimator  structural part                         ML

  Number of observations                            150

Summary Information Measurement + Structural:

  Block Latent Nind Chisq Df
      1      Y    4 1.341  2
      2      M    4 2.319  2
      3      X    4 0.446  2

  Model-based reliability latent variables:

      Y     M     X
  0.745 0.708 0.764

  Summary Information Structural part:
```

```
   chisq df pvalue  cfi rmsea srmr
  13.873  1      0 0.67 0.293 0.11

Parameter Estimates:

  Standard errors                             Twostep
  Information                                Expected
  Information saturated (h1) model          Structured

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  Y ~
    M                 0.429    0.133    3.236    0.001
  M ~
    X                -0.093    0.103   -0.905    0.366

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .Y                 0.781    0.196    3.977    0.000
   .M                 0.815    0.238    3.432    0.001
    X                 0.966    0.237    4.071    0.000
```

last slide

- small samples pose a challenge for SEM estimation and inference
- setting 1: small models, very small sample sizes (nonconvergence, bias)
  - solution: bounded estimation, bias-correction
- setting 2: large models, small sample sizes (misspecifications)
  - solution: structural-after-measurement approach (SAM)
  - for each submodel, we can use a different (consistent) estimator
  - this includes bounded estimation, and/or noniterative estimators
  - many previous solutions turn out to be a special case
- future work:
  - extension to $\mathsf{Var}(\boldsymbol{\eta} \otimes \boldsymbol{\eta})$ (Elissa Burghgraeve)
  - small-sample corrections for SAM (Jasper Bogaert)
  - informative hypothesis testing in small samples (Caroline Keck)
  - noniterative estimators

Thank you!

(questions?)

`http://lavaan.org`